MEAN FLOW POLICY WITH INSTANTANEOUS VELOCITY CONSTRAINT FOR ONE-STEP ACTION GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning expressive and efficient policy functions is a promising direction in reinforcement learning (RL). While flow-based policies have recently proven effective in modeling complex action distributions with a fast deterministic sampling process, they still face a trade-off between expressiveness and computational burden, which is typically controlled by the number of flow steps. In this work, we propose mean flow policy (MFP), a new generative policy function that models the mean velocity field to achieve the fastest one-step action generation. To ensure its high expressiveness, an instantaneous velocity constraint (IVC) is introduced on the mean velocity field during training. We theoretically prove that this design explicitly serves as a crucial boundary condition, thereby improving learning accuracy and enhancing policy expressiveness. Empirically, our MFP achieves state-of-the-art success rates across several challenging robotic manipulation tasks from Robomimic and OGBench. It also delivers substantial improvements in training and inference speed over existing flow-based policy baselines.

1 INTRODUCTION

A promising topic in reinforcement learning (RL) community is to develop expressive and efficient policies, particularly in complex control environments where action distributions can be multi-modal (Zhu et al., 2023; Wang et al., 2023). Generative policies, such as diffusion model and flow matching, have recently emerged as a powerful alternative to Gaussian or mixture policies by transforming simple base distributions into flexible action distributions via learnable transformations (Song et al., 2020; Chi et al., 2023). However, a key limitation of existing generative policies is their dependence on iterative multi-step refinement from noise to actions (Wang et al., 2024a; 2025; Ding et al., 2024). This computational dependency imposes a significant overhead that hinders training speed, particularly for online RL where action sampling is a per-step requirement (Li, 2023; Yang et al., 2023). Moreover, this overhead translates to considerable inference latency, which is a major impediment to achieving high closed-loop performance in real-time control systems (Zhan et al., 2024; 2025; Jiang et al., 2024).

A question naturally arises: *Can we unify the expressiveness of generative policies with the efficiency of one-step action generation for online RL?*

In this paper, we propose the mean flow policy (MFP) as an affirmative answer. While existing flow policies learn instantaneous velocities and require multi-step iterative sampling (Lipman et al., 2023; Park et al., 2025; Bharadhwaj et al., 2024), MFP instead learns the mean velocity field (Geng et al., 2025a). This design enables a direct, single-step mapping from a base Gaussian noise to a multi-modal action distribution, thereby preserving the expressive power of flow-based models while drastically improving training and inference efficiency (Kornilov et al., 2024).

Although the time-efficiency gains of MFP are very promising, its learning difficulty is higher than that

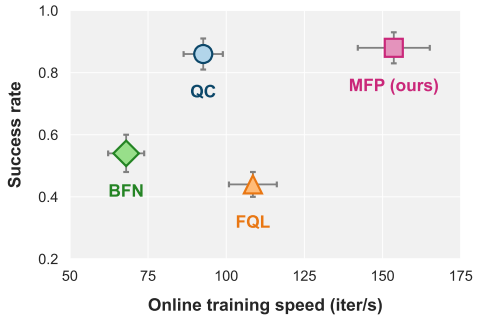


Figure 1: **Performance-efficiency evaluation on 9 robotic manipulation tasks.** Our MFP achieved the highest success rate and fastest training speed on a single A100 GPU.

of a standard flow policy. One reason is that our MFP requires modeling the mean velocity for any time interval specified by two time points (Geng et al., 2025a). A more significant reason is that its learning process is governed by a first-order ordinary differential equation (ODE) derived from the definition of mean velocity. However, this ODE theoretically suffers from the problem of multiple solutions due to a lack of explicit boundary conditions, that is, the value at any boundary point is not enforced. This poses a non-trivial challenge to learning accuracy and consequently affects policy expressiveness (Birkhoff & Langer, 1923).

To address this, we introduce an instantaneous velocity constraint (IVC) to compensate for the lack of boundary conditions. Intuitively, IVC pairs the average velocity loss for each interval with an instantaneous velocity loss at the interval's start point. In practice, IVC is implemented as a auxiliary policy loss, adding negligible computational overhead while materially improving accuracy. We evaluate our MFP on Robomimic (Mandlekar et al., 2021) and OGBench (Park et al., 2024), two demanding robot manipulation benchmarks. As shown in Figure 1 and Table 3, MFP achieves state-of-the-art success rates while delivering substantial speed-ups in training and per-step inference over strong flow-policy baselines, on average across both suites.

Our contributions are summarized threefold:

- We propose a new flow-based policy, namely mean flow policy (MFP), that enables fastest one-step action generation. By modeling the mean velocity field, MFP retains the expressiveness of generative policies while eliminating multi-step sampling overhead.
- We design a training enhancement technique, namely instantaneous velocity constraint (IVC), to improve the learning accuracy of mean velocity field. This technique explicitly serves as a boundary condition, thereby stabilizing learning and enhancing policy expressiveness.
- We empirically achieve state-of-the-art success rates on two challenging robotic manipulation benchmarks: Robomimic and OGBench. Moreover, our approach provides a substantial speedup in both training and inference over existing flow-policy baselines, highlighting its practicality for real-time application.

2 PRELIMINARIES

Reinforcement Learning. We consider an agent interacting with an environment modeled as a Markov Decision Process (MDP), defined by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$ (Li, 2023). The components are the state space $\mathcal{S} \subseteq \mathbb{R}^n$, the action space $\mathcal{A} \subseteq \mathbb{R}^m$, the state transition function $\mathcal{P}(s'|s, a)$, the reward function $r(s, a)$, and the discount factor $\gamma \in [0, 1)$. The primary goal in reinforcement learning (RL) is to learn a policy $\pi(a|s)$ that maximizes the expected cumulative discounted reward, namely return, given by

$$J_\pi = \mathbb{E}_{\pi,\mathcal{P}} \left[\sum_{k=0}^{\infty} \gamma^k r(s_k, a_k) \right]. \tag{1}$$

Grounded in the off-policy learning paradigm, our approach utilizes an action-value function (Q-function) to guide policy improvement, which denotes the expected cumulative return for taking an action a in a state s and thereafter following the policy π.

$$Q^\pi(s, a) = \mathbb{E}_{\pi,\mathcal{P}} \left[\sum_{i=0}^{\infty} \gamma^i r(s_i, a_i) | s_0 = s, a_0 = a \right]. \tag{2}$$

The optimal action-value function, $Q^*(s, a)$, represents the maximum expected return achievable from state s by taking action a. The optimal policy π^* can then be found by selecting the action that maximizes this function: $\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a)$.

Flow Matching. Flow matching is a principled methodology for constructing continuous-time generative models (Lipman et al., 2023). In contrast to diffusion models, which employ stochastic differential equations (SDEs) (Song et al., 2020), flow matching is built upon deterministic dynamics governed by an ordinary differential equation (ODE). By directly learning a continuous-form instantaneous vector field, this approach simplifies the training objective and enables more efficient

sampling. Specifically, it trains a neural network $v_\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ to parameterize a velocity field $v(x(t), t)$ that matches a predefined conditional target velocity $v(1)$.

For a source distribution $q(x(0))$ and a target distribution $p(x(1))$, the velocity field is trained by minimizing the flow matching loss (Lipman et al., 2024):

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{\substack{t\sim\mathcal{U}([0,1]) \\ x(1)\sim p,\, x(0)\sim q}} \left\| v_\theta\big(x(t), t\big) - v(x(t), t) \right\|_2^2, \tag{3}$$

where any intermediate point along the generating path $x(t) = tx(1) + (1-t)x(0)$ is a linear interpolation between source and target points. The velocity for this path is assumed to be a constant vector $v(x(t), t) = x(1) - x(0)$. This formulation defines the target vector field along a straight path between the source and target samples. Once trained, the learned field v_θ defines a probability flow via the following ODE:

$$\frac{dx(t)}{dt} = v_\theta(x(t), t),\ x(0) \sim q, \tag{4}$$

which allows us to effectively generate samples of the target distribution p, starting from samples of the source distribution q. Although flow matching is conceptually designed to generate along a straight line, the paths still curve in practice when fitting between two distributions (Song et al., 2023). Therefore, multi-step discretization and numerical methods like the Euler method are often required to solve the ODE in Eq. (4) to obtain high-quality generated results (Park et al., 2025).

3 METHOD

First, we introduce the mean flow policy (MFP), showing how its integration with a "generate-and-select" mechanism enables a direct mapping from noise to optimal actions. We then present the instantaneous velocity constraint (IVC) and theoretically justify its role in improving the learning accuracy. Finally, the complete pseudo-code for our mean flow RL algorithm is provided.

3.1 MEAN FLOW POLICY

In RL, a policy $\pi(\cdot|s)$ defines a distribution over actions given a state s. For standard flow-based policies, this mapping is framed as a generative process: a velocity model, $v(a(t), t, s)$, transforms a standard Gaussian noise (source) into the optimal action (target), with the state serving as a conditioning input. The final output action $a(1)$ is calculated by

$$a(1) = a(0) + \int_0^1 v(a(\tau), \tau, s)d\tau,\ a(0) \sim \mathcal{N}(0, I). \tag{5}$$

Unlike standard flow policies that learn the instantaneous velocity field $v(a(t), t, s)$, we center on the mean velocity field $u(a(t), t, r, s)$, modeling the mean velocity over any given time interval $[t, r]$ as

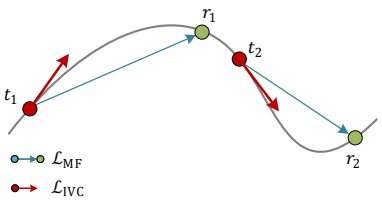


Figure 2: Velocity field: blue arrows denote the mean velocity over a time interval, with red arrows representing the instantaneous velocity at a time point.

$$u(a(t), t, r, s) \triangleq \frac{1}{r-t}\int_t^r v(a(\tau), \tau, s)d\tau. \tag{6}$$

The relationship between these two types of velocities is shown in Figure 2. If the mean velocity model u^* is ideally learned, the policy inference is formulated as

$$a(1) = a(0) + u^*(a(0), 0, 1, s),\ a(0) \sim \mathcal{N}(0, I). \tag{7}$$

(1) How to imitate a given target action. To train the mean velocity model u, we first multiply both sides of Eq. (6) by $(r-t)$ and then differentiate with respect to t (treating r as independent of t), which yields the mean flow identity:

$$-u(a(t), t, r, s) + (r-t)\frac{d}{dt}u(a(t), t, r, s) = -v(a(t), t, s). \tag{8}$$

We assume there is a target optimal action a^*, serving as $a(1)$ for imitation. For two randomly sampled time points, t and r (with $t < r$), the intermediate point $a(t)$ is defined by the linear interpolation: $a(t) = t \cdot a(1) + (1-t) \cdot a(0)$, and $v = a^* - a(0)$. Let θ denote the learnable parameters, the training objective is to minimize the residual of the mean flow identity in Eq. (8) as

$$\mathcal{L}_{\text{MF}}(\theta) = \mathbb{E}_{t,r<t,a(t)} \left\| u_\theta(a(t),t,r,s) - \text{sg}\left(v - (t-r)\frac{d}{dt}u_\theta(a(t),t,r,s)\right)\right\|_2^2, \tag{9}$$

where "sg" denotes the stop-gradient operator to stabilize training. Calculating this training loss requires computing the total time derivative term, $\frac{d}{dt}u$. Using the chain rule, this term is expanded as shown in Eq. (10), and its computation can be performed efficiently using the Jacobian-vector product (JVP) in modern automatic differentiation libraries.

$$\frac{d}{dt}u_\theta(a(t),t,r,s) = v(a(t),t,s) \cdot \frac{\partial}{\partial a}u_\theta(a(t),r,t,s) + \frac{\partial}{\partial t}u_\theta(a(t),r,t,s), \tag{10}$$

By minimizing Eq. (9), the mean velocity model u can effectively transform a standard Gaussian noise into the desired target action distribution. In RL, however, there is no ground-truth dataset of optimal actions to imitate. Next, we will detail how to gradually find better actions and eventually approach the optimal target action.

(2) How to find the optimal target action to imitate. Finding the optimal action a^* directly is not realistic. However, we can use the Q-function to progressively find better actions as imitation targets, and eventually find the optimal action a^* in a bootstrap manner. This mechanism is called "generate-and-select" or "best-of-N".

In practice, at any given state s, the agent first generate N diverse candidate actions as

$$a^i = a^i_k(1) = \epsilon_i + u_\theta(\epsilon_i, 0, 1, s), \;\; a^i(0) = \epsilon^i \sim \mathcal{N}(0, I), \; i = 1, \cdots, N. \tag{11}$$

Then the critic function Q_ϕ parameterized by ϕ is employed to evaluate all candidates, and the action yielding the highest Q-value is identified as the final output action for that state:

$$a^\star = \arg\max_{a^i} Q_\phi(s, a^i). \tag{12}$$

We treat the combined mean flow generation process in Eq. (11) and the best-of-N mechanism in Eq. (12) as a unified policy $\pi_\theta^{\text{unified}}$, or simply denoted as π_θ. The resulting action, $a^\star$, then serves three purposes: (1) interacting with the environment, (2) acting as the target action for policy training, and (3) calculating the target value for value training.

Although this bootstrap mechanism is intuitive, next we will formally prove that such a imitation-based update guarantees policy improvement by the following theorem.

Theorem 1 (Mean Flow Policy Improvement)**.** *Let the new policy π_{new} be derived from an old policy π_{old} via the N-candidate generative update process described above. Under assumptions of a bounded Q-function error (ϵ_Q), L_Q-Lipschitz continuity of the Q-function, and a bounded mean flow matching error (ϵ_A), the performance difference between these two policies is lower-bounded by*

$$V^{\pi_{new}}(s) - V^{\pi_{old}}(s) \geq \underbrace{\mathbb{E}_{\tau\sim\pi_{new}}\left[\sum_{t=0}^{\infty}\gamma^t \Delta_N^{\pi_{old}}(s_t)\right]}_{\textit{BFN improvement term } \Delta_1} - \underbrace{\frac{2\epsilon_Q + L_Q\epsilon_A}{1-\gamma}}_{\textit{fitting error term } \Delta_2}, \tag{13}$$

where $V(s)$ is the state-value function, and $\Delta_N^{\pi_{old}}(s)$ is the best-of-N advantage gain, satisfying

$$\Delta_N^{\pi_{old}}(s) := \mathbb{E}_{a_1,\ldots,a_N\sim\pi_{old}(\cdot|s)}\left[\max_{i=1,\ldots,N} Q^{\pi_{old}}(s,a_i)\right] - V^{\pi_{old}}(s) \geq 0. \tag{14}$$

Proof. See Appendix A. □

This theorem decomposes the performance difference into two distinct components: a *BFN improvement term* Δ_1 and a *fitting error term* Δ_2. As we prove in Appendix A.4, the best-of-N advantage gain ($\Delta_N^{\pi_{\text{old}}}$) in Δ_1 is strictly non-negative, implying that the policy improvement can be driven by the benefit of sampling N diverse action candidates. It is also important to note that this improvement is partially counteracted by the fitting error term Δ_2, which stems from the critic's inaccuracy (ϵ_Q) and the mean flow matching error (ϵ_A). This highlights the importance of reducing ϵ_A to further enhance policy performance. In the next subsection, we will introduce the instantaneous velocity constraint (IVC) to help reduce this fitting error.

3.2 The Instantaneous Velocity Constraint as a Boundary Condition

As we mentioned above, the underlying principle for training the mean velocity model is Eq. (8), which is a first-order ordinary differential equation (ODE) with respect to u. Mathematically speaking, we need to know both the dynamics and at least a boundary condition to accurately solve the unique solution of a ODE. Back to our practical setting, Eq. (8) only provides the dynamics given $t < r$. Although sampling pairs where r is very close to t could implicitly serve as the boundary condition, such events are too rare to ensure robust training.

Inspired by this, we introduce the instantaneous velocity constraint (IVC), a training objective that explicitly enforces a boundary condition at t. The mean velocity from t to t is exactly the known instantaneous velocity $v = a^* - a(0)$, so the IVC objective is expressed as:

$$\mathcal{L}_{\text{IVC}}(\theta) = \mathbb{E}_{t,a(t)} \left\| u_\theta(a(t), t, t) - v \right\|_2^2 . \tag{15}$$

To understand the effectiveness of this IVC objective, we first derive the following theorem, which demonstrates the multiplicity of solutions for Eq. (8) in the absence of a boundary condition.

Theorem 2 (Multiplicity of Solutions for the Mean Flow Identity)**.** *Let the **cumulative error** Δ_u be defined as the difference between the learned and the true mean velocity field, $\Delta_u(a(t), t, r) \triangleq u_\theta(a(t), t, r) - u^*(a(t), t, r)$. Assume u_θ perfectly satisfies the mean flow identity (*Eq. (8)*) for all $t < r$. Then, its cumulative error Δ_u satisfies*

$$\Delta_u(a(t), t, r) = \frac{C(a, r)}{r - t} \tag{16}$$

where $C(a, r)$ is an integration constant independent of time t.

Proof. By assumption, both u_θ and u^* precisely satisfy the mean flow identity. Subtracting the identity for u^* from that of u_θ yields a homogeneous linear differential equation with respect to the cumulative error Δ_u as

$$\Delta_u + (t - r)\frac{d}{dt}\Delta_u = 0. \tag{17}$$

By the product rule for derivatives, this is equivalent to $\frac{d}{dt}[(r - t)\Delta_u] = 0$. Integrating with respect to t yields $(r - t)\Delta_u = C(a, r)$, which completes the proof. □

Theorem 2 reveals that any solution learned by perfectly minimizing the $\mathcal{L}_{\text{MF}}$ loss belongs to a family of functions with a unknown constant $C(a, r)$. Because the $\mathcal{L}_{\text{MF}}$ loss is blind to the boundary, it cannot provide a gradient to force $C(a, r)$ to zero. This allows an arbitrary, persistent bias to exist in the learned u_θ. Next, we derive the following theorem to prove that the explicit introduction of IVC eliminates this degree of freedom and restricts the solution space to the unique correct u^*.

Theorem 3 (Uniqueness via the Instantaneous Velocity Constraint)**.** *Let the **boundary error** Δ_v be defined as the error at the boundary t, $\Delta_v(a(t), t) \triangleq u_\theta(a(t), t, t) - v^*(a(t), t)$. Minimizing the IVC loss, $\mathcal{L}_{IVC} = \mathbb{E}[\|\Delta_v\|^2]$, forces the integration constant $C(a, r)$ in Theorem 2 to zero. This eliminates the boundary error and ensures the cumulative error Δ_u vanishes for all $t < r$.*

Proof. The boundary error Δ_v is the limit of the cumulative error Δ_u as $r \to t^+$. Applying Eq. (16):

$$\Delta_v(a(t), t) = \lim_{r \to t^+} \frac{C(a, r)}{r - t}. \tag{18}$$

This limit diverges if $C(a, r) \neq 0$. To keep the IVC loss finite, the optimization must prevent this divergence, which requires $C(a, r) = 0$. With $C(a, r) = 0$, the boundary error Δ_v is zero. Consequently, by invoking Theorem 2, the cumulative error Δ_u also becomes zero. This completes the proof. □

In essence, Theorem 3 proves that the IVC provides the necessary boundary condition to make the learning problem well-posed. By forcing the error constant to zero at the boundary, the IVC eliminates the cumulative error inherent to the mean flow objective, thereby retaining the high expressive power of the generative model. Moreover, the resulted smaller mean flow matching error ϵ_A in Theorem 1, which helps enable a more effective policy improvement with each update.

3.3 Mean Flow Reinforcement Learning

This section systematically presents the complete picture of our mean flow RL. The policy training loss $\mathcal{L}_{\text{policy}}$ combines the mean velocity model loss in Eq. (9) with the IVC loss in Eq. (15):

$$\mathcal{L}_{\text{policy}}(\theta) = \mathcal{L}_{\text{MF}}(\theta) + \lambda \mathcal{L}_{\text{IVC}}(\theta), \tag{19}$$

where the balancing hyperparameter $\lambda > 0$ is called IVC coefficient, and the default value is 1.0.

Concurrently, the critic Q_ϕ is trained to minimize a standard TD-error in Eq. (20) on transitions (s_k, a_k, r_k, s_{k+1}) from the replay buffer, where k is the step index.

$$\mathcal{L}_Q(\phi) = \mathbb{E}\left[\left(Q_\phi(s_k, a_k) - \left(r_k + \gamma Q_\phi(s_{k+1}, a^\star_{k+1})\right)\right)^2\right]. \tag{20}$$

Recall that we treat the combined mean flow generation process in Eq. (11) and the best-of-N mechanism in Eq. (12) as a unified policy π_θ, so the calculation of $a^\star_{k+1}$ also involves a generative-then-select process for ensuring an unbiased policy evaluation.

The overall training scheme decouples the generative policy's imitative training from the Q-value gradient. Instead, the policy improvement is guaranteed by a best-of-N mechanism under the guidance of Q-value selection. This design allows us to leverage the policy's full expressive power while ensuring stable and effective policy improvement. The complete algorithm is shown in Algorithm 1.

Algorithm 1 Mean Flow RL

Input: Mean Flow policy π_θ, where θ is the parameters of u_θ, Critic Q_ϕ, offline dataset $\mathcal{D}_{\text{offline}}$
Initialize replay buffer $\mathcal{D}$ with $\mathcal{D}_{\text{offline}}$

▷ **Phase 1: Offline Pre-training**

for offline training step **do**
  Replay a mini-batch from $\mathcal{D}$
  Update policy π_θ by minimizing $\mathcal{L}_{\text{policy}}(\theta)$ with Eq. (19)
  Update critic Q_ϕ by minimizing $\mathcal{L}_Q(\phi)$ with Eq. (20)
end for

▷ **Phase 2: Online Interaction and Fine-tuning**

for online training step $k = 1, 2, \ldots$ **do**
  Observe s_k, execute $a^*_k = \pi_\theta(s_k)$, receive s_{k+1}, r_k, and store $(s_k, a^\star_k, r_k, s_{k+1})$ into $\mathcal{D}$
  Replay a mini-batch from $\mathcal{D}$
  Update policy π_θ by minimizing $\mathcal{L}_{\text{policy}}(\theta)$ with Eq. (19)
  Update critic Q_ϕ by minimizing $\mathcal{L}_Q(\phi)$ with Eq. (20)
end for

4 Experiments

4.1 Main Experiment

Benchmark. We consider a total of 9 sparse-reward robotic manipulation tasks with varying difficulties. This includes 3 tasks from the Robomimic benchmark (Mandlekar et al., 2021), `Lift`, `Can` and `Square`, and 6 tasks from OGBench (Park et al., 2024), `cube-double-task 2/3/4` and `cube-triple-task 2/3/4`. For Robomimic, we use the multi-human datasets. For OGBench, we use the default play-style datasets. See more details of these tasks in Appendix C.

Baselines and our method. We compare with three latest strong offline-to-online RL baselines. **(1) FQL (flow Q learning) (Park et al., 2025)** first uses behavioral cloning to train a multi-step flow policy on offline data. It then trains a separate one-step policy that imitates the multi-step policy and maximizes Q-values, enabling efficient and stable learning within the data distribution. **(2) BFN (best-of-N) (Ghasemipour et al., 2021)** combines the best-of-N sampling with an expressive multi-step flow policy. Specifically, BFN first generates N candidate actions and picks the action (out of N) that maximizes the current Q-value. **(3) QC (Q-chunking)** (Li et al., 2025) applies action chunking (Bharadhwaj et al., 2024) on the basis of BFN to improve exploration and sample efficiency. **(4) Ours MFP** also adopts the chunking trick, but leverages a more efficient mean flow policy to achieve the fastest one-step action generation with maintained or even enhanced expressiveness.

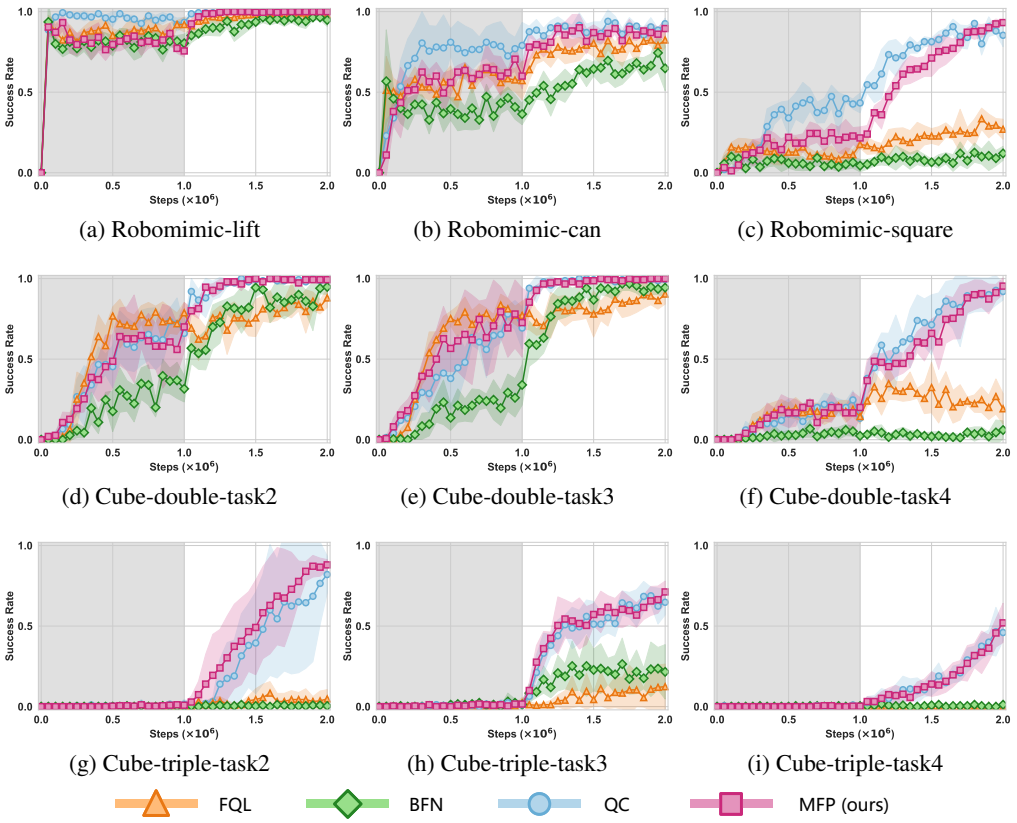


Figure 3: **Training curves on benchmarks.** The solid lines correspond to mean and shaded regions correspond to 95% confidence interval over five runs. The shadow background indicates the offline training phase, while the white background indicates the online training phase.

Table 1: Success rates. Mean ± Std over 5 seeds. **Bold** = best, underlined = 2nd-best.

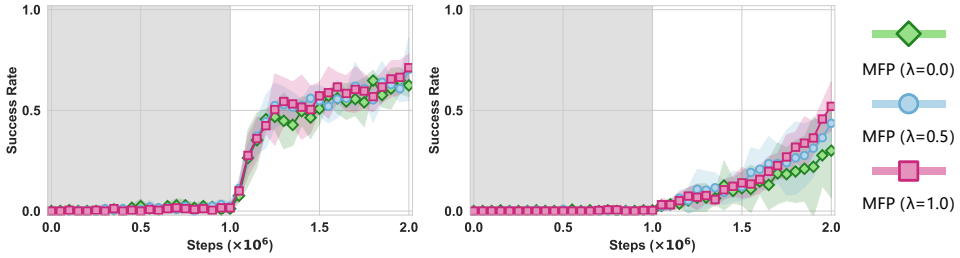
Task	FQL	BFN	QC	MFP (ours)
Robomimic-lift	0.96 ± 0.03	$\mathbf{1.00 \pm 0.01}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
Robomimic-can	0.74 ± 0.11	0.82 ± 0.03	$\mathbf{0.94 \pm 0.06}$	0.92 ± 0.07
Robomimic-square	0.12 ± 0.05	0.34 ± 0.06	0.92 ± 0.01	$\mathbf{0.93 \pm 0.01}$
Cube-double-task2	0.95 ± 0.04	0.88 ± 0.05	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
Cube-double-task3	0.97 ± 0.04	0.90 ± 0.06	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
Cube-double-task4	0.08 ± 0.04	0.35 ± 0.09	0.93 ± 0.08	$\mathbf{0.95 \pm 0.04}$
Cube-triple-task2	0.01 ± 0.02	0.08 ± 0.06	0.82 ± 0.10	$\mathbf{0.88 \pm 0.03}$
Cube-triple-task3	0.12 ± 0.13	0.26 ± 0.14	0.69 ± 0.05	$\mathbf{0.71 \pm 0.06}$
Cube-triple-task4	0.00 ± 0.00	0.02 ± 0.02	0.46 ± 0.13	$\mathbf{0.52 \pm 0.11}$
Average	0.44 ± 0.05	0.52 ± 0.06	0.86 ± 0.05	$\mathbf{0.88 \pm 0.05}$

Main results. As shown in Table 1, our MFP matches or exceeds state-of-the-art multi-step flow-matching baselines on eight of nine tasks. On the remaining task, MFP ranks second, with a performance of 0.92, which is just 0.02 points below the top-performing baseline's score of 0.94. While all methods achieve near-perfect performance on simpler tasks like Robomimic-lift, Cube-double-task2, and Cube-double-task3, MFP demonstrates clear superiority on the more challenging tasks. Specifically, MFP consistently outperforms all baselines on Robomimic-square, Cube-double-task4, and all Cube-triple tasks, where it consistently achieves the highest success rates. For instance, on the most difficult task, Cube-triple-task4, MFP achieves a success rate of 0.52 ± 0.11, which is

significantly higher than the next-best baseline, QC (0.46 ± 0.13), and substantially exceeds both FQL and BFN. Overall, our MFP secures the top position with an average success rate of 0.88 ± 0.05. This result highlights its strong capability that is competitive with multi-step flow policies in solving long-horizon, sparse-reward tasks.

4.2 Ablation Study and Supplementary Experiments

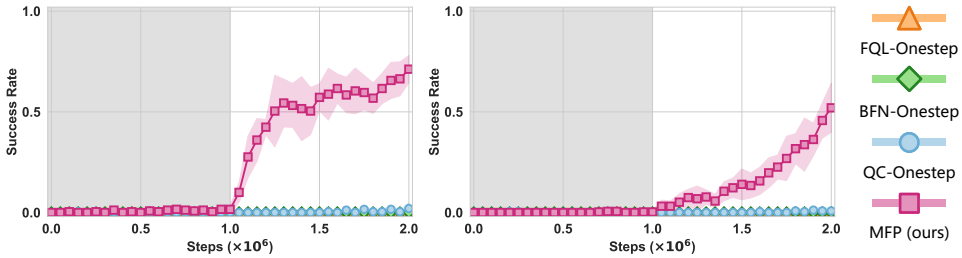
(1) Ablation on the instantaneous velocity constraint (IVC). We perform an ablation study on the IVC coefficient λ. Our full version ($\lambda = 1.0$) was compared against variants with a reduced constraint ($\lambda = 0.5$) and without the constraint ($\lambda = 0.0$). The results, as shown in Figure 4, indicate a positive correlation between the IVC weight and performance, while also demonstrating that the method is not overly sensitive to λ. For example, the success rate on the challenging Cube-triple-task4 significantly increases from 0.30 ± 0.21 (with no IVC) to 0.45 ± 0.15 (with a partial IVC), and further to 0.52 ± 0.11 (with full IVC). Detailed numerical results are listed in Table 4 in Appendix B.1. These findings empirically validate our theoretical claims, confirming the IVC's role as a crucial component for modeling an accurate mean velocity field and consequently achieving significant performance gains.



(a) Cube-triple-task3 (b) Cube-triple-task4

Figure 4: **Training curves of ablation on the IVC.**

(2) Comparison with one-step variants of the aforementioned baselines. We compared our MFP against one-step variants of the aforementioned baselines: FQL-Onestep, BFN-Onestep, and QC-Onestep. As shown in Figure 5, a naive one-step configuration is insufficient for solving these complex, long-horizon tasks, with baselines achieving success rates near zero on both Cube-triple-task3 and Cube-triple-task4. In stark contrast, our MFP achieves success rates of 0.71 ± 0.06 and 0.52 ± 0.11 on these tasks, respectively. Detailed numerical results can be seen in Table 5 in Appendix B.1. This supplementary comparison highlights that simply using a one-step standard flow is not enough; the superior expressive capability and stable learning process of our mean flow policy are critical for tackling these challenging long-horizon manipulation tasks.



(a) Cube-triple-task3 (b) Cube-triple-task4

Figure 5: **Training curves of comparison with one-step flow.**

(3) Training and inference time analysis. Figure 1 presents a comparison of the average success rate (%) versus online training speed (iter/s) across the all 9 tasks. Our MFP achieves highest success rate and fastest training speed. This superior training efficiency stems directly from our

one-step action generation, which eliminates the expensive iterative sampling process required by prior multi-step flow policies.

Table 2: Comparison of online training speed

Task	FQL	BFN	QC	MFP (ours)
Average	108.5 ± 7.7 iter/s	68.0 ± 5.8 iter/s	92.6 ± 6.3 iter/s	153.6 ± 11.5 iter/s

Regarding inference time-efficiency, we conducted an evaluation with a focus on its suitability for real-world robotic deployment. Since robotic platforms often have limited computational resources, our experiments were conducted on a CPU-only environment, AMD Ryzen Threadripper 3960X 24-Core Processor. To simulate a more realistic deployment scenario without hardware acceleration, we disabled JAX's Just-In-Time (JIT) compilation during all evaluations.

The results are listed in Table 3. our MFP and FQL exhibit very similar inference times, with both approaches being significantly faster than BFN and QC.

Why BFN and QC are slow. The poor performance of BFN and QC is primarily because they rely on a 10-step flow policy, which requires iterative computation to transform noise into an action.

Why FQL is fast. FQL simultaneously learns both a 10-step flow policy for high-accuracy imitation and a separate one-step flow policy. The one-step policy is obtained through a distillation process combined with a Q-function maximization loss, which allows FQL to achieve a high inference speed comparable to our MFP.

Why MFP is better than FQL comprehensively. Despite FQL's relatively fast inference, its training process is very slow due to the involvement of multiple policies, including a multi-step flow policy. Furthermore, benchmark results indicate that its success rate is very low, averaging only half of our MFP's. When considering FQL's overall low success rate and slow training speed, our MFP still maintains a significant advantage.

Table 3: Comparison of inference time

Task	FQL	BFN	QC	MFP (ours)
Average	10.76 ± 1.02 ms	117.3 ± 13.23 ms	113.22 ± 11.92 ms	10.93 ± 0.95 ms

4.3 Supplementary Results

We also extended our evaluation to high-dimensional D4RL tasks and visual manipulation tasks, and conduct a series of ablation results to demonstrate MFP's superior performance across diverse situations. Please see Appendix E and F for full details.

5 Related Work

Offline-to-online RL. The offline-to-online RL first uses a static, pre-collected dataset to offline train initial policy and Q-value functions (Levine et al., 2020; Kumar et al., 2020). This process provides a warm start, giving the agent a foundational understanding of the environment that significantly accelerates and improves the efficiency of subsequent online interactive fine-tuning (Lee et al., 2022). Numerous algorithmic designs have been proposed to improve offline-to-online RL, including behavioral regularization (Ashvin et al., 2020; Tarasov et al., 2023), conservatism (Kumar et al., 2020), in-sample maximization (Kostrikov et al., 2022; Garg et al., 2023), out-of-distribution detection (Yu et al., 2020; Kidambi et al., 2020; Nikulin et al., 2023) and dual RL (Lee et al., 2021; Sikchi et al., 2024). Beyond these algorithmic solutions, the choice of policy function also plays an important role (Wang et al., 2023). A policy network with high expressiveness can better capture the intricate distribution of the behavioral policy during offline training stage. This capability is also crucial for

adapting to target environments during online fine-tuning, as optimal actions in these settings often possess a naturally multi-modal nature (Park et al., 2025; Li et al., 2025). Since the policy must infer actions at every step during both the online fine-tuning and real-time deployment stages, the inference time efficiency of the policy function is also critical (Li, 2023; Park et al., 2025). Our work contributes a new policy function, MFP, which achieves the fastest single-step action generation and maintains a high expressive capacity to achieve high performance.

Generative models as RL policies. The expressive power of generative models, such as denoising diffusion models (Ho et al., 2020) and flow matching (Lipman et al., 2022), makes them promising for representing complex, multi-modal policies in both offline (Janner et al., 2022; Chi et al., 2023) and online (Yang et al., 2023; Wang et al., 2024a; Ding et al., 2024) RL. However, their iterative sampling process requires a high number of function evaluations (NFE), creating a prohibitive latency for the high-throughput nature of online RL. A common approach in RL to improving time-efficiency of generative policies is distillation, which compresses a trained iterative model into a one-step policy (Wang et al., 2024b; Park et al., 2025). Beyond the field of RL, recent studies have begun to explore training a single-step generative flow directly for tasks like image generation (Lu & Song, 2025). These methods typically operate either by enforcing consistency constraints on the model's outputs at different time steps (Song et al., 2023; Song & Dhariwal, 2024; Geng et al., 2025b) or by explicitly modeling the flow velocity over a specific time interval (Boffi et al., 2024; Frans et al., 2024). The latter class of methods generally requires more iterations to train, but ultimately performs better (Frans et al., 2024). A typical representative, mean flow, has achieved the best one-step fitting performance on Imagenet (Geng et al., 2025a). We propose a new policy function, MFP, which combines mean flow with RL to enable the fastest single-step action generation. In the context of RL, the dynamic shifts in data distribution during sampling place higher demands on imitation-based flow matching training. To address this, our proposed IVC serves as an explicit boundary condition during MFP training, which reduces fitting error and consequently ensures strong policy expressiveness.

6 Conclusion

We propose the mean flow policy (MFP), a new generative RL policy that enjoys both high time-efficiency and expressiveness. The former stems from the fastest one-step action generation, and the latter contributes to our designed instantaneous velocity constraint (IVC), which explicitly serves as a necessary boundary condition and reliably improves the learning accuracy. Empirical results on the Robomimic and OGBench benchmarks confirm that our MFP achieves state-of-the-art success rates and offers substantial improvements in training and inference speed. We believe that this work represents a significant step towards developing highly expressive and efficient policy functions for complex robotic control tasks. Regarding limitation, a primary one is the additional GPU memory consumption during training, which stems from the Jacobian-Vector Product (JVP) operation. In future work, we plan to validate our method on more tasks and real robotic platforms.

References

Nair Ashvin, Dalal Murtaza, Gupta Abhishek, and L Sergey. Accelerating online reinforcement learning with offline datasets. *CoRR, vol. abs/2006.09359*, 2020.

Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4788–4795. IEEE, 2024.

George D Birkhoff and Rudolph E Langer. The boundary problems and developments associated with a system of ordinary linear differential equations of the first order. In *Proceedings of the American Academy of Arts and Sciences*, volume 58, pp. 51–128. JSTOR, 1923.

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. Flow map matching. *arXiv preprint arXiv:2406.07507*, 2, 2024.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, pp. 02783649241273668, 2023.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. In *The Eleventh International Conference on Learning Representations*, 2023.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling, 2025a. URL https://arxiv.org/abs/2505.13447.

Zhengyang Geng, Ashwini Pokle, Weijian Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pp. 3682–3691. PMLR, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

Yuxuan Jiang, Yujie Yang, Zhiqian Lan, Guojian Zhan, Shengbo Eben Li, Qi Sun, Jian Ma, Tianwen Yu, and Changwu Zhang. Rocket landing control with random annealing jump start reinforcement learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 14026–14033. IEEE, 2024.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Nikita Kornilov, Petr Mokrov, Alexander Gasnikov, and Aleksandr Korotin. Optimal flow matching: Learning straight trajectories in just one step. *Advances in Neural Information Processing Systems*, 37:104180–104204, 2024.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Jongmin Lee, Wonseok Jeon, Byungjun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In *International Conference on Machine Learning*, pp. 6120–6130. PMLR, 2021.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic Q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. *arXiv preprint arXiv:2507.07969*, 2025.

Shengbo Eben Li. *Reinforcement Learning for Sequential Decision and Optimal Control*. Springer Verlag, Singapore, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

Alexander Nikulin, Vladislav Kurenkov, Denis Tarasov, and Sergey Kolesnikov. Anti-exploration by random network distillation. In *International conference on machine learning*, pp. 26228–26244. PMLR, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. *ArXiv*, 2024.

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

H Sikchi, A Zhang, and S Niekum. Dual rl: Unification and new methods for reinforcement and imitation learning. In *International Conference on Learning Representations*. International Conference on Learning Representations, 2024.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252. PMLR, 2023.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:11592–11620, 2023.

Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, et al. Diffusion actor-critic with entropy regulator. *Advances in Neural Information Processing Systems*, 37:54183–54204, 2024a.

Yinuo Wang, Mining Tan, Wenjun Zou, Haotian Lin, Xujie Song, Wenxuan Wang, Tong Liu, Likun Wang, Guojian Zhan, Tianze Zhu, et al. Enhanced dacer algorithm with high diffusion efficiency. *arXiv preprint arXiv:2505.23426*, 2025.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024b.

Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Guojian Zhan, Yuxuan Jiang, Shengbo Eben Li, Yao Lyu, Xiangteng Zhang, and Yuming Yin. A transformation-aggregation framework for state representation of autonomous driving systems. *IEEE Transactions on Intelligent Transportation Systems*, 25(7):7311–7322, 2024.

Guojian Zhan, Xin An, Yuxuan Jiang, Jingliang Duan, Huichan Zhao, and Shengbo Eben Li. Physics informed neural pose estimation for real-time shape reconstruction of soft continuum robots. *IEEE Robotics and Automation Letters*, 2025.

Zhengbang Zhu, Hanye Zhao, Haoran He, Yichao Zhong, Shenyu Zhang, Haoquan Guo, Tingting Chen, and Weinan Zhang. Diffusion models for reinforcement learning: A survey. *arXiv preprint arXiv:2311.01223*, 2023.

A THEORETICAL ANALYSIS ON THE MEAN FLOW POLICY IMPROVEMENT

In this section, we provide the detailed theoretical analysis for the policy improvement guarantee of our training paradigm. It includes the implementation procedures of policy update detailed in A.1, core assumptions and useful lemma in A.2, the formal proof of the mean flow policy improvement theorem in A.3, and three key properties of the improvement gain in A.4.

A.1 IMPLEMENTATION PROCEDURES OF POLICY UPDATE

Let π_{old} denote the base mean flow policy prior to an update. At the same time, we have a learned critic, $Q_\phi(s, a)$, which is an approximation of the true action-value function of the base policy, $Q^{\pi_{\text{old}}}(s, a)$. Let π_{new} denote the updated policy, which is derived from π_{old} through the following three-step generative process for any given state s:

1. **Sample**: Generate N candidate actions $\{a_1, \ldots, a_N\}$ by sampling from the base policy, i.e., $a_i \sim \pi_{\text{old}}(\cdot|s)$.
2. **Select**: Use the learned critic Q_ϕ to select the best action from the candidates, which becomes the target action $a^*(s)$:
$$a^*(s) = \arg\max_{a_i} Q_\phi(s, a_i).$$
3. **Match**: The new policy π_{new} is obtained by training the parameters of π_{old} to match the target action $a^*(s)$. Note that an action $a_{\text{new}} \sim \pi_{\text{new}}$ is not guaranteed to perfectly match $a^*(s)$. We formally bound the expected distance between them in Assumption 3.

For analytical clarity, we introduce a conditional matching distribution, $M(\cdot|a^*)$, to model the outcome of the matching process (Step 3). Specifically, we assume the specific target a^* has been selected in the previous Steps 1 and 2, then generation of the final action a_{new} is expressed as

$$a_{\text{new}} \sim M(\cdot|a^*).$$

The difference between $M(\cdot|a^*)$ and π_{new} is that $M(\cdot|a^*)$ describes the action distribution conditioned on a specific target action a^*, whereas π_{new} represents the marginal distribution of the action, which results from averaging over all possible targets $a^*(s)$ generated in Steps 1 and 2.

A.2 CORE ASSUMPTIONS AND USEFUL LEMMA

Our analysis relies on the following assumptions and lemma.

Assumption 1 (Bounded Q-Value Fitting Error)**.** *The error between the learned critic Q_ϕ and the true Q-function of the base policy $Q^{\pi_{old}}$ is uniformly bounded by a constant $\epsilon_Q \geq 0$.*

$$\forall (s, a) \in \mathcal{S} \times \mathcal{A}, \quad |Q_\phi(s, a) - Q^{\pi_{old}}(s, a)| \leq \epsilon_Q.$$

Assumption 2 (Q-Value Smoothness)**.** *The true Q-value of the base policy, $Q^{\pi_{old}}(s, a)$, is L_Q-Lipschitz continuous with respect to the action a.*

$$\forall s \in \mathcal{S},\ a_1, a_2 \in \mathcal{A}, \quad |Q^{\pi_{old}}(s, a_1) - Q^{\pi_{old}}(s, a_2)| \leq L_Q \cdot d(a_1, a_2).$$

Assumption 3 (Bounded Mean Flow Matching Error)**.** *The expected distance between the action a_{new} generated by the new policy and the target action a^* is bounded by a constant $\epsilon_A \geq 0$.*

$$\forall s \in \mathcal{S}, \quad \mathbb{E}_{a_{new} \sim M(\cdot|a^*)}[d(a_{new}, a^*(s))] \leq \epsilon_A.$$

Lemma 1 (Performance Difference Lemma (Kakade & Langford, 2002))**.** *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ be a Markov Decision Process, and let π and π' be two arbitrary policies. The difference in the state-value function at a starting state s, or denoted as s_0, can be expressed in terms of the advantage function of the new policy π' with respect to the old policy π as:*

$$V^{\pi'}(s) - V^{\pi}(s) = \mathbb{E}_{s \sim d^{\pi'}, a \sim \pi'(\cdot|s)} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s, a) \right].$$

where $d^{\pi'}(s)$ is the discounted state visitation distribution under policy π', and $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$ is the advantage function of policy π.

Proof. The proof relies on the key identity $V^{\pi}(s) = \sum_{a\in\mathcal{A}} \pi(a|s) Q^{\pi}(s,a)$. We start with the definition of the value difference for a state s:

$$\begin{aligned} V^{\pi'}(s) - V^{\pi}(s) &= V^{\pi'}(s) - \sum_{a\in\mathcal{A}} \pi'(a|s) Q^{\pi}(s,a) + \sum_{a\in\mathcal{A}} \pi'(a|s) Q^{\pi}(s,a) - V^{\pi}(s) \\ &= V^{\pi'}(s) - \sum_{a\in\mathcal{A}} \pi'(a|s) Q^{\pi}(s,a) + \sum_{a\in\mathcal{A}} \pi'(a|s) (Q^{\pi}(s,a) - V^{\pi}(s)) \\ &= V^{\pi'}(s) - \sum_{a\in\mathcal{A}} \pi'(a|s) Q^{\pi}(s,a) + \sum_{a\in\mathcal{A}} \pi'(a|s) A^{\pi}(s,a) \end{aligned}$$

Using the Bellman expectation equation for $V^{\pi'}(s)$:

$$V^{\pi'}(s) = \sum_{a'\in\mathcal{A}} \pi'(a'|s) \left(R(s,a') + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a') V^{\pi'}(s') \right)$$

And the definition of $Q^{\pi}(s,a)$:

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a) V^{\pi}(s')$$

Substitute these into our first equation:

$$\begin{aligned} & V^{\pi'}(s) - V^{\pi}(s) \\ &= \sum_{a'\in\mathcal{A}} \pi'(a'|s) \left(R(s,a') + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a') V^{\pi'}(s') - Q^{\pi}(s,a') \right) + \sum_{a'\in\mathcal{A}} \pi'(a'|s) A^{\pi}(s,a') \\ &= \sum_{a'\in\mathcal{A}} \pi'(a'|s) \left(\gamma \sum_{s'\in\mathcal{S}} P(s'|s,a') (V^{\pi'}(s') - V^{\pi}(s')) \right) + \mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)] \\ &= \gamma \mathbb{E}_{a\sim\pi'(\cdot|s), s'\sim P(\cdot|s,a)}[V^{\pi'}(s') - V^{\pi}(s')] + \mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)] \end{aligned}$$

This recursive formula shows that the value difference at state s depends on the expected value difference at the next state s'. By recursively unrolling this expression over time and averaging over the state visitation distribution $d^{\pi'}(s)$, we arrive at the final result.

$$V^{\pi'}(s) - V^{\pi}(s) = \mathbb{E}_{s\sim d^{\pi'}, a\sim\pi'(\cdot|s)} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s,a) \right].$$

This completes the proof. □

A.3 Proof of the mean flow policy improvement theorem

This subsection provides a formal proof of the mean flow policy improvement theorem, that is, Theorem 1 in the main text.

Proof. Our proof begins by invoking the Performance Difference Lemma, i.e., Lemma 1, which yields

$$V^{\pi_{\text{new}}}(s) - V^{\pi_{\text{old}}}(s) = \mathbb{E}_{\tau\sim\pi_{\text{new}}} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_{\text{old}}}(s_t, a_t) \right]. \tag{21}$$

This equation connects the improvement in value function to the expected advantage of the new policy's actions, measured with respect to the old policy.

The following proof proceeds in two parts: first, we derive the single-step advantage bound $\mathbb{E}_{a_t\sim\pi_{\text{new}}(\cdot|s_t)}[A^{\pi_{\text{old}}}(s_t,a_t)]$, and second, we aggregate them over time to obtain the required bound of value difference, which completes the proof.

Part 1: Lower bound the single-step expected advantage We begin by analyzing the Q-value decay due to the imperfect matching for a *given* target action a^*. The final action a_{new} is distributed according to $M(\cdot|a^*)$. We can bound the expected difference as follows:

$$\begin{aligned}
&Q^{\pi_{\text{old}}}(s,a^*) - \mathbb{E}_{a_{\text{new}}\sim M(\cdot|a^*)}[Q^{\pi_{\text{old}}}(s,a_{\text{new}})] \\
&\quad = \mathbb{E}_{a_{\text{new}}}[Q^{\pi_{\text{old}}}(s,a^*) - Q^{\pi_{\text{old}}}(s,a_{\text{new}})] \\
&\quad \le \mathbb{E}_{a_{\text{new}}}[|Q^{\pi_{\text{old}}}(s,a^*) - Q^{\pi_{\text{old}}}(s,a_{\text{new}})|] && \text{(by Jensen's inequality)} \\
&\quad \le \mathbb{E}_{a_{\text{new}}}[L_Q \cdot d(a^*, a_{\text{new}})] && \text{(by Q-Value Smoothness, Assumption. 2)} \\
&\quad \le L_Q \epsilon_A && \text{(by Bounded Matching Error, Assumption 3)}
\end{aligned}$$

Rearranging gives us the following bound for a given a^*:

$$\mathbb{E}_{a_{\text{new}}\sim M(\cdot|a^*)}[Q^{\pi_{\text{old}}}(s,a_{\text{new}})] \ge Q^{\pi_{\text{old}}}(s,a^*) - L_Q\epsilon_A.$$

Now, we take the expectation over the distribution of target actions $a^*(s)$ to get the bound for the marginal policy π_{new}:

$$\begin{aligned}
\mathbb{E}_{a\sim\pi_{\text{new}}}[Q^{\pi_{\text{old}}}(s,a)] &= \mathbb{E}_{\{a_i\}\sim\pi_{\text{old}}}\left[\mathbb{E}_{a_{\text{new}}\sim M(\cdot|a^*(s))}[Q^{\pi_{\text{old}}}(s,a_{\text{new}})]\right] \\
&\ge \mathbb{E}_{\{a_i\}\sim\pi_{\text{old}}}[Q^{\pi_{\text{old}}}(s,a^*)] - L_Q\epsilon_A.
\end{aligned}$$

Next, we relate the Q-value of the target action, $Q^{\pi_{\text{old}}}(s,a^*)$, back to the Q-values of the original candidates $\{a_i\}$:

$$\begin{aligned}
Q^{\pi_{\text{old}}}(s,a^*) &\ge Q_\phi(s,a^*) - \epsilon_Q && \text{(by Bounded Q-Value Fitting Error, Assumption 1)} \\
&= \max_{i=1,\dots,N} Q_\phi(s,a_i) - \epsilon_Q && \text{(by definition of } a^*) \\
&\ge \max_{i=1,\dots,N} Q^{\pi_{\text{old}}}(s,a_i) - 2\epsilon_Q. && \text{(by Assumption 1 on each candidate } a_i)
\end{aligned}$$

Substituting this result back into our bound for $\mathbb{E}_{a\sim\pi_{\text{new}}}[Q^{\pi_{\text{old}}}(s,a)]$, we get:

$$\mathbb{E}_{a\sim\pi_{\text{new}}}[Q^{\pi_{\text{old}}}(s,a)] \ge \mathbb{E}_{\{a_i\}\sim\pi_{\text{old}}}\left[\max_{i=1,\dots,N} Q^{\pi_{\text{old}}}(s,a_i)\right] - 2\epsilon_Q - L_Q\epsilon_A.$$

Finally, subtracting $V^{\pi_{\text{old}}}(s) = \mathbb{E}_{a\sim\pi_{\text{old}}}[Q^{\pi_{\text{old}}}(s,a)]$ from both sides gives the desired single-step advantage bound:

$$\mathbb{E}_{a\sim\pi_{\text{new}}}[A^{\pi_{\text{old}}}(s,a)] \ge \Delta_N^{\pi_{\text{old}}}(s) - 2\epsilon_Q - L_Q\epsilon_A,$$

where $\Delta_N^{\pi_{\text{old}}}(s)$ is the **best-of-N advantage gain**, defined as

$$\Delta_N^{\pi_{\text{old}}}(s) := \mathbb{E}_{a_1,\dots,a_N\sim\pi_{\text{old}}(\cdot|s)}\left[\max_{i=1,\dots,N} Q^{\pi_{\text{old}}}(s,a_i)\right] - V^{\pi_{\text{old}}}(s).$$

Part 2: Extension to the value function difference With the single-step advantage bound established, we substitute it into Eq. (21) and obtain:

$$\begin{aligned}
&V^{\pi_{\text{new}}}(s) - V^{\pi_{\text{old}}}(s) \\
&\quad = \mathbb{E}_{\tau\sim\pi_{\text{new}}}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi_{\text{old}}}(s_t,a_t)\right] \\
&\quad = \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{s_t\sim d^t_{s,\pi_{\text{new}}}}\left[\mathbb{E}_{a_t\sim\pi_{\text{new}}(\cdot|s_t)}[A^{\pi_{\text{old}}}(s_t,a_t)]\right] && \text{(by law of total expectation)} \\
&\quad \ge \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{s_t\sim d^t_{s,\pi_{\text{new}}}}\left[\Delta_N^{\pi_{\text{old}}}(s_t) - 2\epsilon_Q - L_Q\epsilon_A\right] && \text{(by substituting the bound from Part 1)} \\
&\quad = \mathbb{E}_{\tau\sim\pi_{\text{new}}}\left[\sum_{t=0}^{\infty}\gamma^t \Delta_N^{\pi_{\text{old}}}(s_t)\right] - \frac{2\epsilon_Q + L_Q\epsilon_A}{1-\gamma}. && \text{(by rearranging the geometric series)}
\end{aligned}$$

This completes the proof. □

A.4 KEY PROPERTIES OF THE BEST-OF-N ADVANTAGE GAIN

The term $\Delta_N^{\pi_{\text{old}}}(s)$ has several important properties:

1. **Non-negativity**: $\Delta_N^{\pi_{\text{old}}}(s) \geq 0$.
2. **Monotonicity with** N: $\Delta_{N+1}^{\pi_{\text{old}}}(s) \geq \Delta_N^{\pi_{\text{old}}}(s)$.
3. **Special case** $(N = 1)$: $\Delta_1^{\pi_{\text{old}}}(s) = 0$.

Proof. Our proof sketch is to first rewrite $\Delta_N^{\pi_{\text{old}}}(s)$ as an integral involving the cumulative distribution function (CDF) of the Q-values, and then demonstrate the claimed properties of the rewritten form.

Let us define a random variable X representing the candidate Q-values. For a given state s, the randomness of X is induced by the candidate action a independently sampled from the base policy:

$$X = Q^{\pi_{\text{old}}}(s, a), \quad \text{where } a \sim \pi_{\text{old}}(\cdot|s).$$

Recall that the definition of the **best-of-N advantage gain** is

$$\Delta_N^{\pi_{\text{old}}}(s) := \mathbb{E}_{a_1,\ldots,a_N \sim \pi_{\text{old}}(\cdot|s)} \left[\max_{i=1,\ldots,N} Q^{\pi_{\text{old}}}(s, a_i) \right] - V^{\pi_{\text{old}}}(s).$$

We can equivalently describe this definition using the CDF notation of the random variable X. Let $F_X(x)$ be the CDF of X, and $X_1, \ldots, X_N$ be N independent and identically distributed (i.i.d.) samples of X, and let $Y_N = \max(X_1, \ldots, X_N)$. The CDF of Y_N is $F_{Y_N}(y) = [F_X(y)]^N$. Therefore, we have $\Delta_N^{\pi_{\text{old}}}(s) = \mathbb{E}[Y_N] - \mathbb{E}[X]$.

Moving forward, since the expectation of a random variable X in terms of its CDF satisfies $\mathbb{E}[X] = \int_0^\infty (1 - F_X(x))dx - \int_{-\infty}^0 F_X(x)dx$, we can further rewrite $\Delta_N^{\pi_{\text{old}}}(s)$ as:

$$\begin{aligned}\Delta_N^{\pi_{\text{old}}}(s) &= \left(\int_0^\infty (1 - F_{Y_N}(x))dx - \int_{-\infty}^0 F_{Y_N}(x)dx \right) - \left(\int_0^\infty (1 - F_X(x))dx - \int_{-\infty}^0 F_X(x)dx \right) \\ &= \int_{-\infty}^\infty (F_X(x) - F_{Y_N}(x))dx.\end{aligned}$$

Substituting $F_{Y_N}(x) = [F_X(x)]^N$, we arrive at the final rewritten form:

$$\Delta_N^{\pi_{\text{old}}}(s) = \int_{-\infty}^\infty (F_X(x) - [F_X(x)]^N)dx.$$

Next we use this form to prove the three key properties of $\Delta_N^{\pi_{\text{old}}}(s)$.

1. Proof of non-negativity ($\Delta_N^{\pi_{\text{old}}}(s) \geq 0$)

The CDF $F_X(x)$ takes values in the range $[0, 1]$. For any value $p \in [0, 1]$ and any integer $N \geq 1$, we have $p^N \leq p$. Therefore, the integrand $F_X(x) - [F_X(x)]^N$ is always greater than or equal to zero for all x. The integral of a non-negative function is non-negative. Thus, $\Delta_N^{\pi_{\text{old}}}(s) \geq 0$.

2. Proof of monotonicity with N ($\Delta_{N+1}^{\pi_{\text{old}}}(s) \geq \Delta_N^{\pi_{\text{old}}}(s)$)

We examine the difference between consecutive terms using the integral form:

$$\begin{aligned}\Delta_{N+1}^{\pi_{\text{old}}}(s) - \Delta_N^{\pi_{\text{old}}}(s) &= \int_{-\infty}^\infty (F_X(x) - [F_X(x)]^{N+1})dx - \int_{-\infty}^\infty (F_X(x) - [F_X(x)]^N)dx \\ &= \int_{-\infty}^\infty ([F_X(x)]^N - [F_X(x)]^{N+1})dx \\ &= \int_{-\infty}^\infty [F_X(x)]^N (1 - F_X(x))dx.\end{aligned}$$

Since $F_X(x) \in [0, 1]$, both terms in the integrand, $[F_X(x)]^N$ and $(1 - F_X(x))$, are non-negative. Their product is therefore non-negative. The integral of this non-negative function is non-negative, which implies $\Delta_{N+1}^{\pi_{\text{old}}}(s) - \Delta_N^{\pi_{\text{old}}}(s) \geq 0$.

3. Proof of special case ($\Delta_1^{\pi_{\text{old}}}(s) = 0$)

Substituting $N = 1$ into the integral identity yields:

$$\Delta_1^{\pi_{\text{old}}}(s) = \int_{-\infty}^{\infty} (F_X(x) - [F_X(x)]^1) dx = \int_{-\infty}^{\infty} (F_X(x) - F_X(x)) dx = \int_{-\infty}^{\infty} 0 \, dx = 0.$$

This confirms the special case. □

B Supplementary Results

B.1 Numerical results of ablation study

Table 4: Ablation on the impact of IVC.

Task	**MFP** ($\lambda = 0.0$)	**MFP** ($\lambda = 0.5$)	**MFP** ($\lambda = 1.0$)
Cube-triple-task3	0.65 ± 0.05	0.70 ± 0.14	$\mathbf{0.71 \pm 0.06}$
Cube-triple-task4	0.30 ± 0.21	0.44 ± 0.08	$\mathbf{0.52 \pm 0.11}$

Table 5: Comparison with one-step variants of the aforementioned baselines.

Task	FQL-Onestep	BFN-Onestep	QC-Onestep	MFP (ours)
Cube-triple-task3	0.00 ± 0.01	0.00 ± 0.00	0.02 ± 0.03	$\mathbf{0.71 \pm 0.06}$
Cube-triple-task4	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	$\mathbf{0.52 \pm 0.11}$

C Environments Description

Robomimic benchmark. We use three challenging tasks from the robomimic domain (Mandlekar et al., 2021). We use the multi-human datasets that were collected by six human operators. Each dataset consists of 50 trajectories provided by each operator, for a total of 300 successful trajectories. The operators were varied in proficiency – there were 2 "worse" operators, 2 "okay" operators, and 2 "better" operators, resulting in diverse, mixed quality datasets. The three tasks are as described below.

(1) lift: requires the robot arm to pick a small cube. This is the simplest task of the benchmark.

(2) can: requires the robot arm to pick up a coke can and place in a smaller container bin.

(3) square: requires the robot arm to pick a square nut and place it on a rod. The nut is slightly bigger than the rod and requires the arm to move precisely to complete the task successfully.

All of the three robomimic tasks use binary task completion rewards where the agent receives -1 reward when the task is not completed and 0 reward when the task is completed.

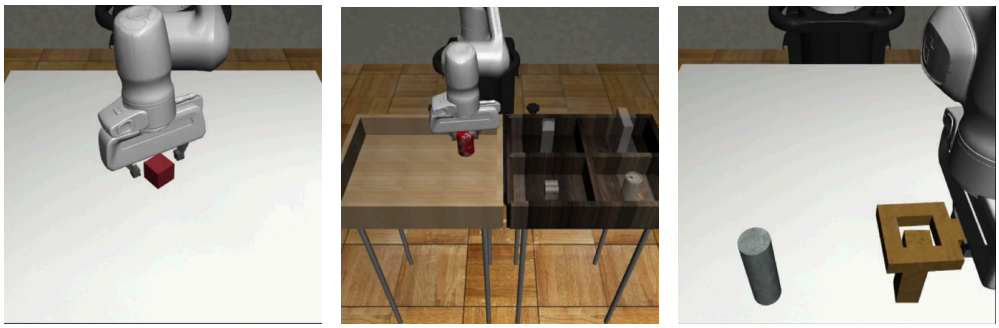
OGBench `cube-double/triple`: These three domains contain 2/3 cubes respectively. The tasks in the two domains all involve moving the cubes to their desired locations. The reward is $-n_{\text{wrong}}$ where n_{wrong} is the number of the cubes that are at the wrong position. The episode terminates when all cubes are at the correct position (reward is 0).

Below we highlight three representative tasks from each environment:

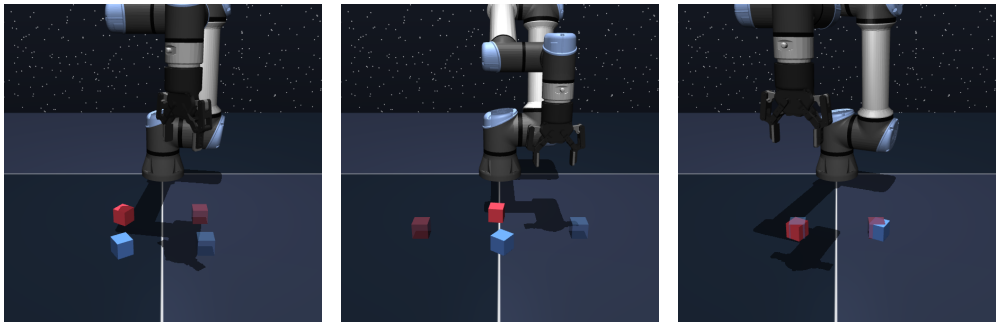
(4) Cube-double-task2 (move): Two cubes in different colors are initialized at $(0.35, -0.1, 0.02)$ and $(0.50, -0.1, 0.02)$, and the goal is to move them to $(0.35, 0.1, 0.02)$ and $(0.50, 0.1, 0.02)$.

(5) Cube-double-task3 (move): The initial cube positions are $(0.35, 0.0, 0.02)$ and $(0.50, 0.0, 0.02)$, and they must be placed at $(0.425, -0.2, 0.02)$ and $(0.425, 0.2, 0.02)$.

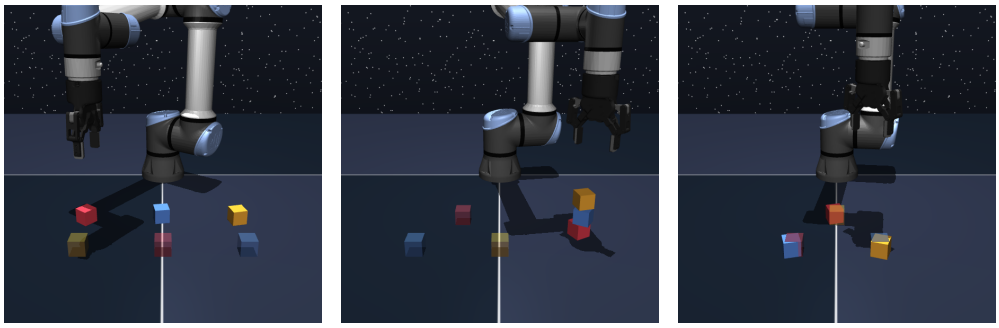
(6) Cube-double-task4 (swap): Two cubes start from $(0.425, -0.1, 0.02)$ and $(0.425, 0.1, 0.02)$, and the objective is to swap their positions to $(0.425, 0.1, 0.02)$ and $(0.425, -0.1, 0.02)$.



(a) Lift (b) Can (c) Square



(d) Cube-double-task2 (e) Cube-double-task3 (f) Cube-double-task4



(g) Cube-triple-task2 (h) Cube-triple-task3 (i) Cube-triple-task4

Figure 6: **Snapshots of the 9 challenging long-horizon, sparse-reward manipulation tasks.**

(7) Cube-triple-task2 (move): Three cubes are initialized at $(0.35, -0.2, 0.02)$, $(0.35, 0.0, 0.02)$, and $(0.35, 0.2, 0.02)$, with goals at $(0.50, 0.0, 0.02)$, $(0.50, 0.2, 0.02)$, and $(0.50, -0.2, 0.02)$.

(8) Cube-triple-task3 (stack): A stacked tower of three cubes is initialized at $(0.425, 0.2, 0.02)$, $(0.425, 0.2, 0.06)$, and $(0.425, 0.2, 0.10)$, and the robot must relocate them to $(0.35, -0.1, 0.02)$, $(0.50, -0.2, 0.02)$, and $(0.50, 0.0, 0.02)$, respectively.

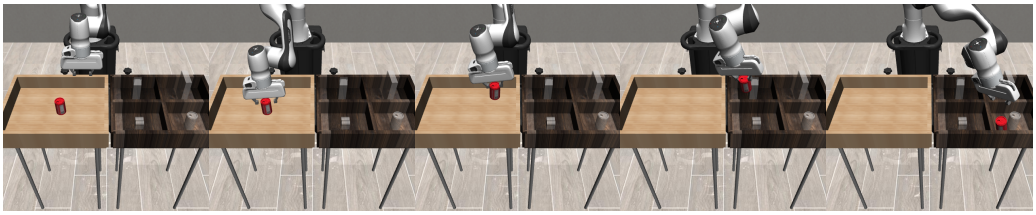
(9) Cube-triple-task4 (swap): Three cubes are initialized at $(0.35, 0.0, 0.02)$, $(0.50, -0.1, 0.02)$, and $(0.50, 0.1, 0.02)$, and they must be cyclically rearranged to $(0.50, -0.1, 0.02)$, $(0.50, 0.1, 0.02)$, and $(0.35, 0.0, 0.02)$.

These tasks highlight increasingly complex spatial reasoning requirements, ranging from coordinated multi-cube relocation to disassembly of stacked structures and cyclic rearrangement.

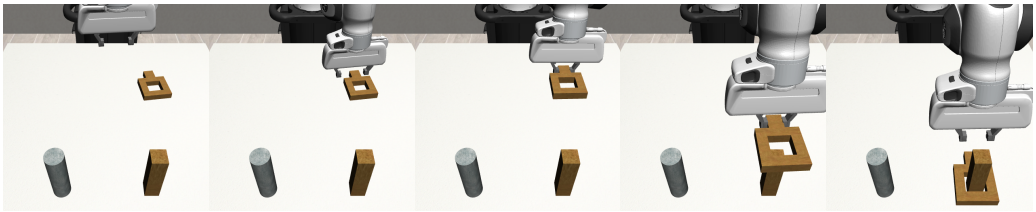
D VISUALIZATIONS

This section provides supplementary visual material to demonstrate our model's performance. The included visualizations of successful episodes highlight our policy's ability to generate precise and robust trajectories, showcasing its effectiveness in handling the complexities of **long-horizon reasoning** and **sparse rewards** inherent in these robotic manipulation tasks.

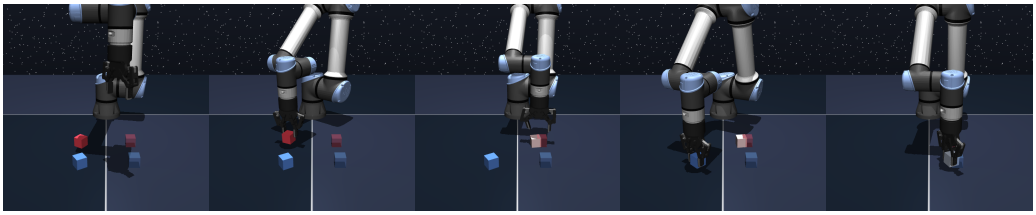
(a) **Lift** (b) $step = 5$ (c) $step = 10$ (d) $step = 15$ (e) $step = 20$



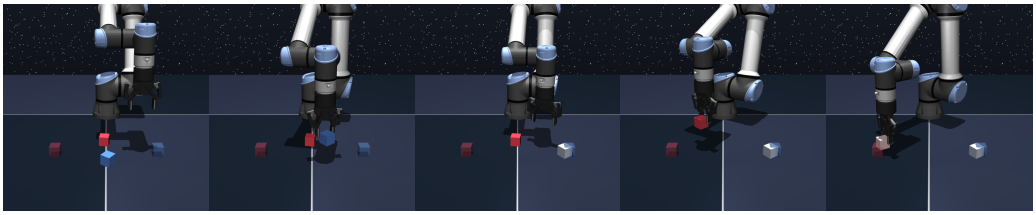
(f) **Can** (g) $step = 15$ (h) $step = 30$ (i) $step = 45$ (j) $step = 65$



(k) **Square** (l) $step = 20$ (m) $step = 40$ (n) $step = 60$ (o) $step = 70$

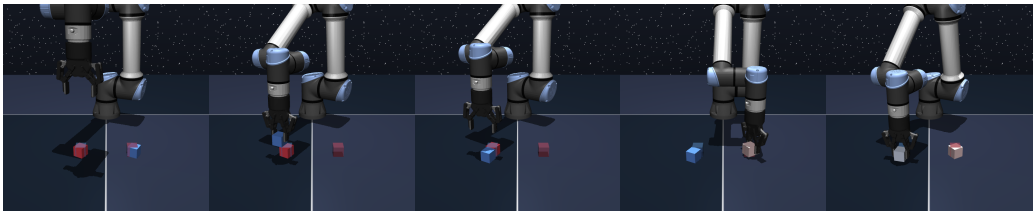


(p) **Double-task2** (q) $step = 15$ (r) $step = 30$ (s) $step = 45$ (t) $step = 60$

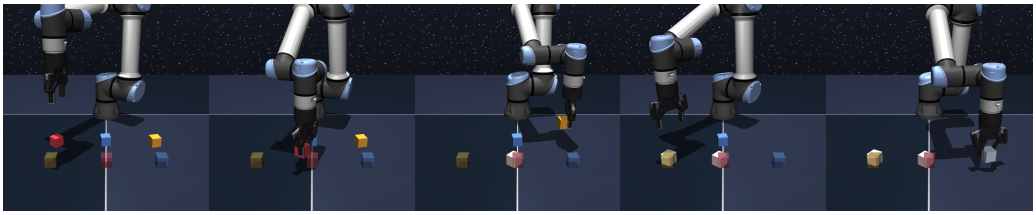


(u) **Triple-task2** (v) $step = 20$ (w) $step = 30$ (x) $step = 45$ (y) $step = 50$

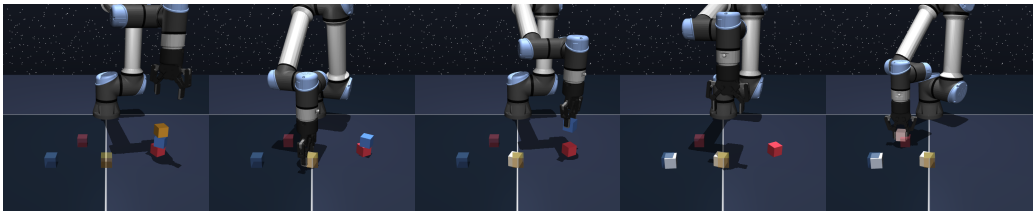
Figure 7: **Visualizations of typical success episodes: Robomimic-lift, Robomimic-can, Robomimic-square, Cube-double-task2, and Cube-double-task3.**



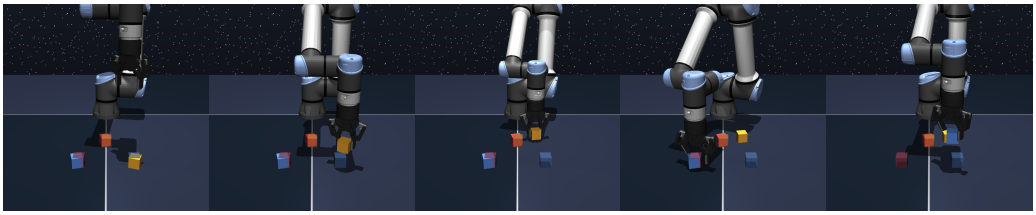
(a) **Double-task4** (b) $step = 40$ (c) $step = 75$ (d) $step = 120$ (e) $step = 155$



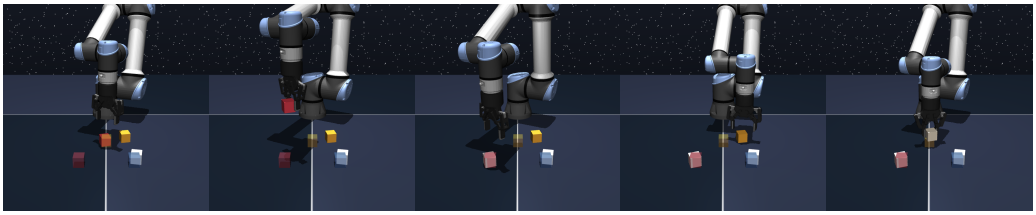
(f) **Triple-task2** (g) $step = 30$ (h) $step = 60$ (i) $step = 90$ (j) $step = 125$



(k) **Triple-task3** (l) $step = 40$ (m) $step = 75$ (n) $step = 120$ (o) $step = 165$



(p) **Triple-task4** (q) $step = 25$ (r) $step = 50$ (s) $step = 75$ (t) $step = 100$



(u) $step = 125$ (v) $step = 150$ (w) $step = 175$ (x) $step = 200$ (y) $step = 235$

Figure 8: **Visualizations of typical success episodes: Cube-double-task4, Cube-triple-task2, Cube-triple-task3, and Cube-triple-task4.**

E ADDITIONAL EXPERIMENTS ON HIGH-DIMENSIONAL AND VISUAL TASKS

To verify the scalability and robustness of MFP across diverse domains, we extended our evaluation to include two high-dimensional control tasks and two visual manipulation tasks, as shown in 9.

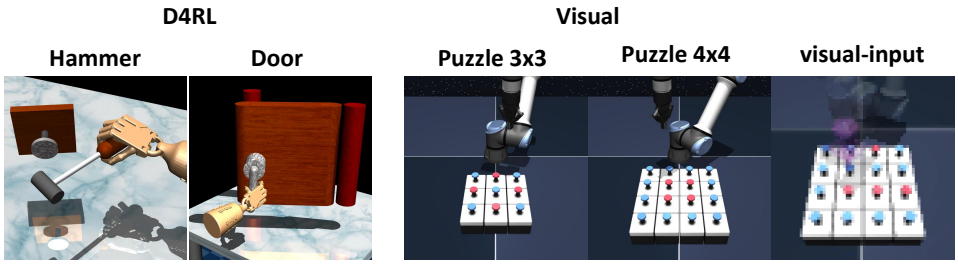


Figure 9: Snapshots of the additional tasks.

E.1 HIGH-DIMENSIONAL CONTROL (D4RL ADROIT)

We evaluated MFP on the D4RL Adroit suite (Hammer and Door), which involves high-dimensional dexterous manipulation. As shown in Table 6 and Figure 10, MFP significantly outperforms the strong baseline QC, achieving 14 points increase in normalized scores on average. This highlights MFP's superior capability in modeling complex action distributions in high-dimensional spaces.

Table 6: Comparison of normalized scores on D4RL Adroit tasks (Mean $\pm$ Std).

Algorithm	D4RL-Door	D4RL-Hammer
QC	51.8 ± 6.0	92.2 ± 4.5
MFP (ours)	$\mathbf{64.7 \pm 4.7}$	$\mathbf{108.3 \pm 5.2}$

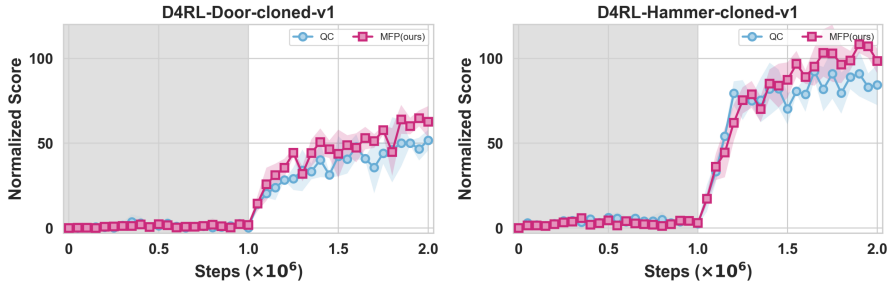


Figure 10: Training curves on D4RL Adroit tasks.

E.2 VISUAL-BASED MANIPULATION

We further evaluated MFP on visual-based Puzzle-3x3 and Puzzle-4x4 tasks, which require spatial reasoning directly from pixel inputs. The results are summarized in Table 7. While both methods solve the 3x3 task, MFP achieves a higher success rate (0.61 vs. 0.47) and better peak performance on the significantly harder 4x4 task, demonstrating its effectiveness in learning visual policies with sparse rewards.

Table 7: Success rates on Visual-based manipulation tasks.

Algorithm	Visual-puzzle-3x3	Visual-puzzle-4x4
QC	1.0 ± 0.0	0.47 ± 0.33
MFP (ours)	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.61 \pm 0.40}$

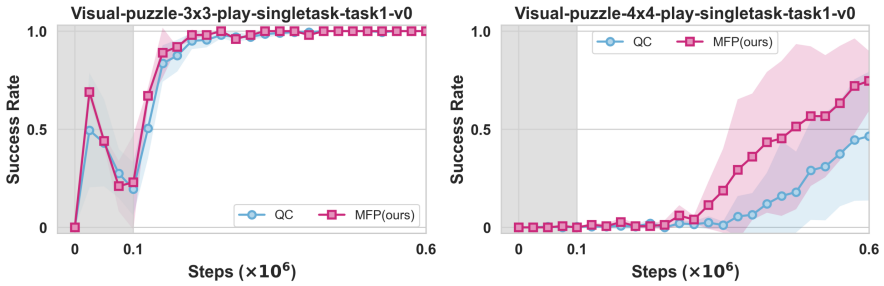


Figure 11: Training curves on visual tasks.

F Additional Ablation Studies

F.1 Impact of Inference Steps

To demonstrate the trade-off between efficiency and performance, we evaluated the baseline QC with varying inference steps (1, 5, and 20). As shown in Figure 12 and Table 8, reducing the baseline steps leads to a catastrophic performance drop. In contrast, MFP is structurally designed for one-step generation (NFE=1) and effectively dominates the Pareto frontier, achieving SOTA success rates with minimal computational cost.

Table 8: Performance comparison with varying inference steps.

Algorithm	Cube-triple-task3	Cube-triple-task4
QC (1 step)	0.02 ± 0.02	0.01 ± 0.91
QC (5 steps)	0.33 ± 0.40	0.08 ± 0.04
QC (20 steps)	0.69 ± 0.07	0.30 ± 0.08
MFP (ours)	$\mathbf{0.71 \pm 0.08}$	$\mathbf{0.52 \pm 0.11}$

F.2 Comparison with Probabilistic Mixing

We compared our proposed Instantaneous Velocity Constraint (IVC) against the heuristic probabilistic mixing strategy used in prior MeanFlow work (Geng et al., 2025a). As shown in Figure 13, our simultaneous IVC loss yields significantly higher success rates and stability compared to mixing strategies (25%, 50%, 75%), confirming that explicit boundary supervision is crucial for RL tasks.

G Evaluation on Imitation Learning

To verify the generality of our method, we conducted a pure Behavioral Cloning (BC) experiment on the Robomimic Square task. Our one-step MFP outperforms the multi-step baseline in success rate while maintaining significantly faster inference. Additionally, visualization on the Push-T task (Figure 15) confirms that MFP successfully captures multi-modal behaviors.

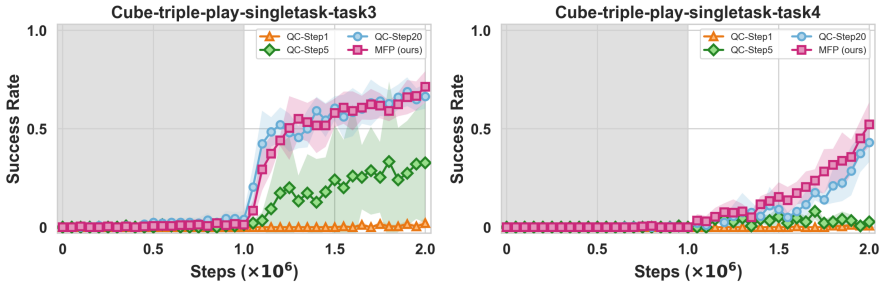


Figure 12: Ablation study on flow inference steps.

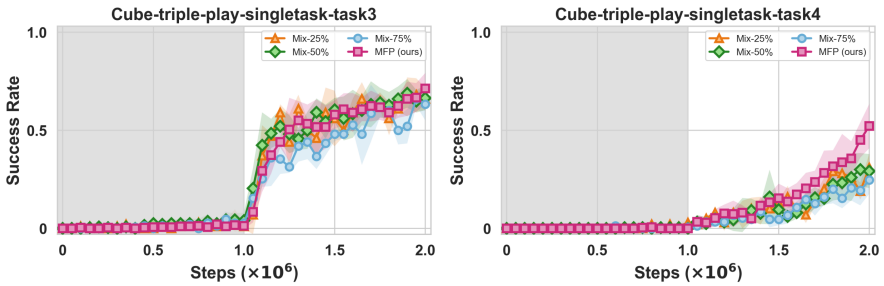


Figure 13: Ablation study comparing IVC with probabilistic mixing strategies.

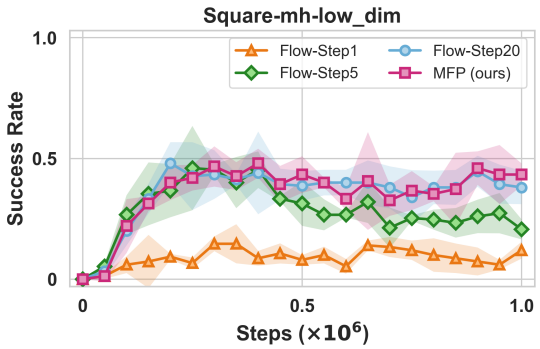


Figure 14: Behavior cloning experiment.

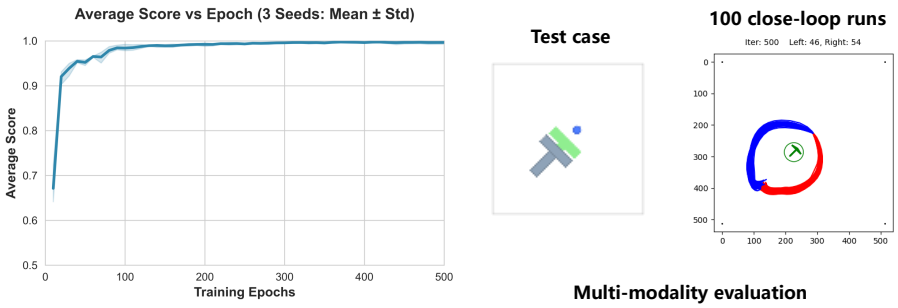


Figure 15: Visualization of multi-modal behaviors learned by MFP on the Push-T task.

H REPRODUCIBILITY STATEMENT

The hyperparameters of all algorithms are shown in Table 9.

Parameter	Value
Shared	
Batch size	256
Discount factor (γ)	0.99
Optimizer	Adam
Learning rate	3×10^{-4}
Target network update rate (τ)	5×10^{-3}
UTD Ratio	1
Evaluation interval	5000
Number of evaluation episodes	50
Number of offline training steps	1×10^{6} (1M)
Number of online training steps	1×10^{6} (1M)
Number of flow steps (T)	10
Policy network width	512
Policy network depth	4 hidden layers
Policy activation function	GELU
Policy layer normalization	False
Value network width	512
Value network depth	4 hidden layers
Value activation function	GELU
Value layer normalization	True
Value ensemble size (K)	2
Value ensemble operator	MEAN
FQL	
Flow step	10
BC weight (α)	10000 for `lift`, `can` and `square` 300 for `cube-double-*` and `cube-triple-*`
QC	
Chunking horizon length	5
Flow step	10
Number of best-of-N	16 for `lift`, `can` and `square` 32 for `cube-double-*` and `cube-triple-*`
MFP-IVC (ours)	
IVC ratio (λ)	1.0
Number of best-of-N	16 for `lift`, `can` and `square` 32 for `cube-double-*` and `cube-triple-*`

Table 9: **Detailed hyperparameters.**

I LLM USAGE DISCLOSURE

We used ChatGPT to refine the grammar and improve the clarity of the text. All LLM-generated suggestions were reviewed and edited by the authors, who take full responsibility for the final content.